intel®

ARTICLE REPRINT

AR-113

January 1980

LSI Chips Ease Standard 488 Bus Interfacing

Ronald M. Williams
Intel Corporation, Santa Clara, California

Reprinted with permission of *Computer Design Magazine*, October 1979 Issue.

231320-001

LSI CHIPS EASE STANDARD 488 BUS INTERFACING

Time and cost disadvantages of interfacing to the IEEE Std 488 bus are overcome with a dedicated LSI chip set that incorporates most of its functional and electrical specifications

Ronald M. Williams Intel Corporation, Santa Clara, California

Historically, interface techniques proliferated as designers evolved customized links among instruments, controllers, and processors for realtime test measurements or data communications, resulting in excessive and expensive codes, formats, signal levels, and timing factors. Obviously, interface standardization was mandatory to save design costs for engineers, development costs for manufacturers, and system integration costs for users. Thus, IEEE Standard 488-1978 (a revision of ANSI/IEEE Std 488-1975) offers a universal instrumentation system approach to automatic operating measurement configurations that provides compatibility, versatility, and flexibility. This system approach establishes a suitable standard bus for interfacing programmable devices from different manufacturers. Outstanding advantages of the standard bus include byte serial, bit parallel digital data handling, synchronized communication among devices at varying data rates, and hardware interchangeability and interconnection in daisy-chained fashion. However, some restrictive disadvantages that have hindered implementation are highly complex logic protocol, time consuming design analysis, and lack of low cost components to perform the intricate logic control functions. To overcome these drawbacks, a large scale integrated (LSI) chip set has been designed with built-in IEEE Std 488 logic controls. Thus,

231320-001

interfacing has been significantly simplified for properly connecting processor buses and programming system protocols.

Interface Overview

The IEEE Standard 488-1978 bus interface includes electrical, mechanical, and functional specifications* for interconnecting both programmable and nonprogrammable electronic measuring apparatus with other apparatus and accessories necessary to assemble instrumentation systems. The functional specifications occupy about 80% of the document and involve a proportional amount of system design time to implement. Bus functions encompass 16 active signal lines, 10 interface functions, the protocol by which interface functions send and receive messages, and logical and timing relationships between signal states.

Functional requirements of the standard can be incorporated in either hardware, software, or a combination of both. Some designers have chosen the hardware approach to incorporate all the interface functions, using about 200 medium scale integrated (MSI) and small scale integrated (SSI) packages. This technique costs about $1000 for a complete interface board. As a result, many cost sensitive implementations of the bus interface use only a subset of its functions custom tailored to the requirements of the devices involved, thereby reducing package count and expense by curtailing the interchangeability advantages.

Other designers have selected the software approach to implement the bus interface. One disadvantage of this approach is that programming is an expensive and extended project; another is that a subroutine has to be executed with each transferred byte. This overhead not only burdens the microprocessor within a device, but also reduces the overall speed of the bus. This approach costs about $200 for the interfacing functions.

*This article deals with the functional aspects (interface signals that exist on the physical bus) of IEEE Std 488-1978, and is not intended as a complete dissertation on the major elements of the standard. For detailed definitions of the mechanical (physical cable connections), electrical (timing, voltages, and currents), and operational (application software routines) technicalities, interested readers should consult the *IEEE Standard Digital Interface for Programmable Instrumentation*, IEEE Std 488-1978, Institute of Electrical and Electronics Engineers, Inc, New York, NY 10017, Nov 30, 1978—**Ed.**

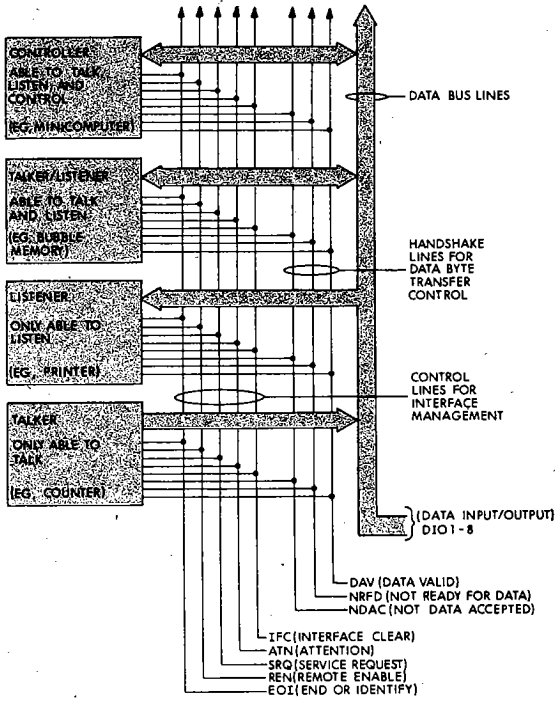


Fig 1 IEEE Std 488 active signal lines for multiple devices. Peripheral devices of different characteristics can be easily connected to standard bus interface. Controller (or processor), such as minicomputer, enables and disables talkers and listeners and manages overall bus activity. Bubble memory functions as both talker and listener. As listener, printer receives characters to be printed. As talker, counter transmits measurements to both controller and listeners

231320-001

Combinational hardware/software approaches, although faster than direct software implementations, still require enormous design time and cost about $1000 for a typical interface board.

With a recent alternative approach, however, the bus interface is easier and less expensive to incorporate in instrument designs. LSI circuit chips now include as built-in capabilities most of the functional and some of the electrical portions of the Standard's specifications, significantly reducing design time and costing about $50 for bus interfacing. Additionally, Intel's 8291/8292 General Purpose Interface Bus (GPIB) peripheral chip set also incorporates capabilities for bus monitoring, data rate manipulation, and addressing to further simplify bus interface designs.

Bus Signal Definitions

The IEEE Std 488 signals are defined as negative true, where the high state (0 = false, ≥2.0 V) and the low state (1 = true, ≤0.8 V) are based on standard transistor-transistor logic (TTL) levels. Of the 16 active signal lines, 8 are data lines, 5 are interface management lines, and 3 are handshake lines (Fig 1). Data input/output lines (DIO1-DIO8) carry ASCII-coded information, as well as device addresses, universal commands, or program instructions. Interface management lines help to supervise the data lines. The primary management line—Attention (ATN)—determines how data lines are processed. When ATN is true, data lines are interpreted as addresses or universal commands by all bus connected devices. When ATN is false, only those devices addressed can use the data lines; in this case, data transmitted are typically device-dependent. With another management line, Interface Clear (IFC), the bus controller returns the system to a known quiescent state. The Service Request (SRQ) line can be used by any device on the interface bus when it has data to send (talker) or needs to receive data (listener). The Remote Enable (REN) line determines whether the system is under front panel or program control. The End Or Identify (EOI) line can be used as a delimiter by a talker (sending) device to indicate an end of message, or by the controller as a polling line.

Handshake lines control the timing relationship of the interface bus (Fig 2). The Data Valid (DAV) line

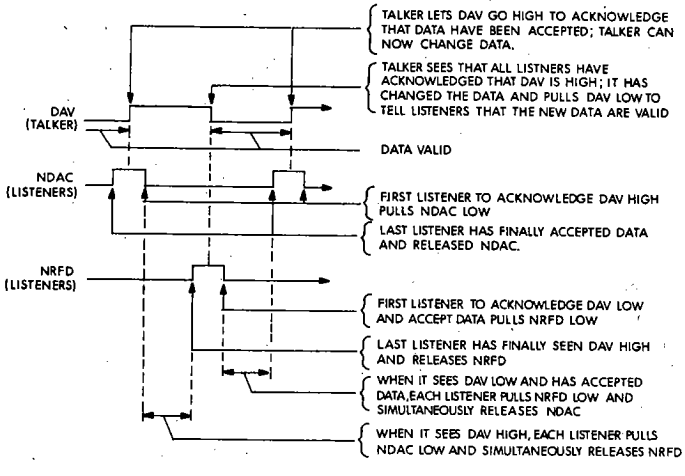


Fig 2 Three-wire handshaking between single talker and several listeners. Before transfer begins, listener indicates it is ready by asserting Ready For Data (RFD) message to true. Talker then drives all eight data input/output lines. Following settling time specified by standard, talker asserts Data Valid (DAV) message to true. While data are being read, RFD message is asserted to false since device is unable to receive additional data. As each listener completes its read, it indicates acceptance by asserting Data Accepted (DAC) message to true; DAC is not sensed true by talker until all listeners have completed read. After each device indicates acceptance, it indicates readiness for data by asserting RFD to true. New cycle begins when all devices have asserted RFD to true

231320-001

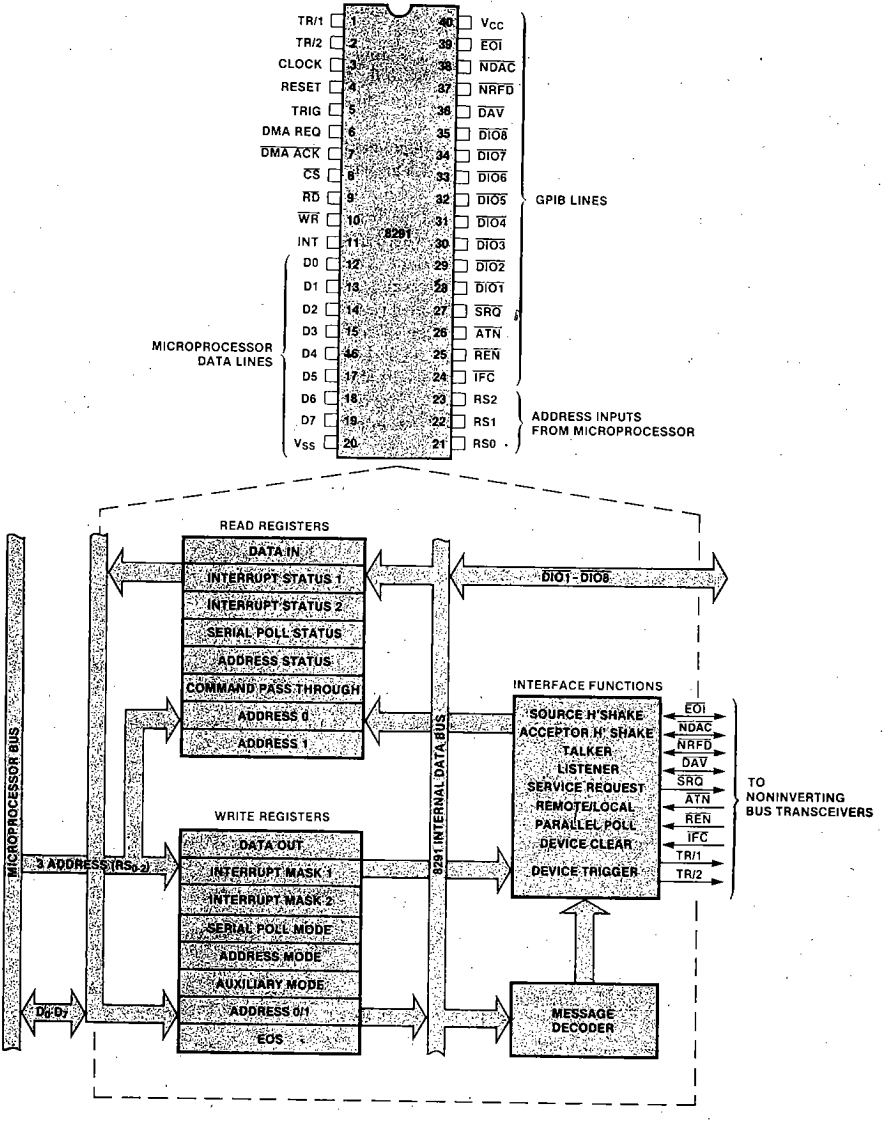
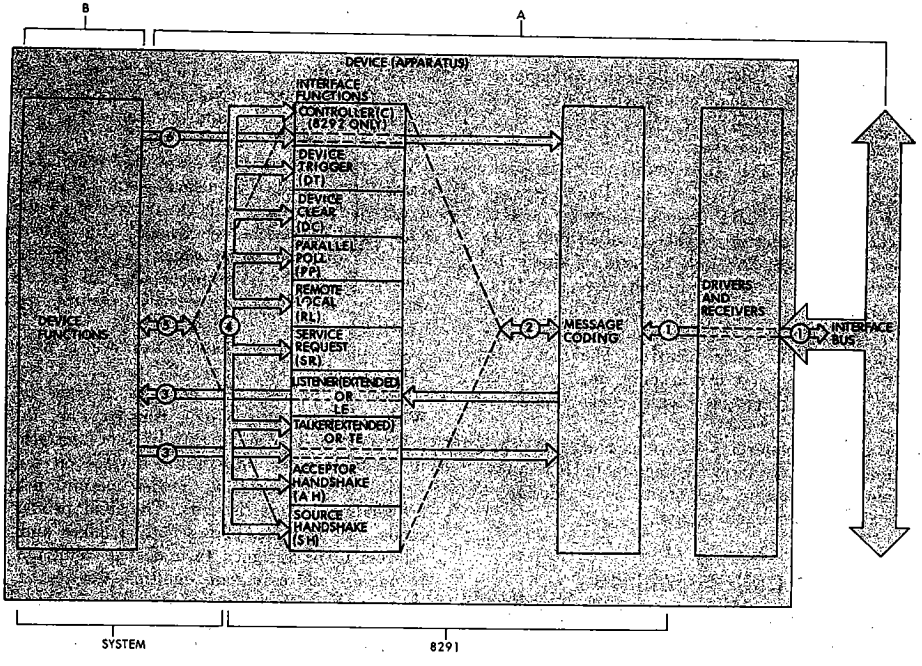


Fig 3 GPIB talker/listener chip. 8291 chip connects 8-bit microprocessor to noninverting bus transceivers, which, in turn, connect to IEEE Std 488 bus. Microprocessor manipulates data bytes after receipt or before transmission, and monitors talker/listener status. Single chip handles all IEEE Std 488 interface functions, except controller functions



A - CAPABILITY DEFINED BY 488-1978 STANDARD
B - CAPABILITY DEFINED BY DESIGNER
1 - INTERFACE BUS SIGNAL LINES
2 - REMOTE INTERFACE MESSAGES TO AND FROM INTERFACE FUNCTIONS
3 - DEVICE DEPENDENT MESSAGES TO AND FROM DEVICE FUNCTIONS
4 - STATE LINKAGES BETWEEN INTERFACE FUNCTIONS
5 - LOCAL MESSAGES BETWEEN DEVICE FUNCTIONS AND INTERFACE FUNCTIONS (MESSAGES TO INTERFACE FUNCTIONS ARE DEFINED; MESSAGES FROM INTERFACE FUNCTIONS EXIST ACCORDING TO DESIGNER
6 - REMOTE INTERFACE MESSAGES SENT BY DEVICE FUNCTIONS WITHIN CONTROLLER (8292)

Fig 4 Bus interface functions. Messages received from interface bus can cause state transitions, just as state transitions can cause messages to be sent on bus (1 and 2). Device dependent data are transferred automatically to microprocessor, without affecting state transitions (3). State changes in one function can cause state changes in another function, resulting in message to be sent (4). Microprocessor can also send local messages to interface functions (5) or remote messages to interface (6)

is used by a talker device to indicate that data are ready to transmit. The Not Ready For Data (NRFD) and Not Data Accepted (NDAC) lines are used by a listener to indicate readiness to receive data and receipt of data, respectively. As a result, a talker knows when all listeners on the bus have received an 8-bit byte of information. Thus, the transmission rate of the bus is only as fast as the slowest listener.

Messages conveyed by all 16 lines are true or false, depending on the states of 10 interface functions. The standard defines each of these interface functions with state diagrams. A function's state can be changed by a controller, another device on the bus, or a state change in another function within a device. Of the 10 interface functions, four provide basic communication capabilities: Source Handshake (SH), Talker (T), Acceptor Handshake (AH), and Listener (L). These functions affect the three handshake lines (DAV, NRFD, and NDAC), eight data lines (DIO1-DIO8), and EOI management line. The Device Clear (DC) and Device Trigger (DT) interface functions are used to initialize and to trigger a device, respectively. The Parallel Poll (PP) function acts with the EOI line to send a single bit of status information. The Service Request (SRQ) function controls the SRQ management line. The Remote Local (RL) interface uses the REN management line in conjunction with front panel control. The Controller (C) function, which is active in only one device on the bus at a time, determines which device talks or listens.

To date, these 10 interface functions and their intricate interrelationship and timing factors have required difficult and time consuming efforts when designing the interface bus into a digital system.

Talker/Listener Chip Capabilities

The 8291 GPIB talker/listener chip, a 40-pin LSI device (Fig 3), performs the inversion necessary to connect an 8-bit microprocessor bus to the negative true IEEE Std 488 bus. In addition, this chip implements most of the Standard's required functions. The microprocessor sets the talker/listener chip to an initial state, manipulates bytes before or after transmission, performs interrupt service routines, causes state changes, monitors other state changes, and enables and disables chip capabilities.

Without microprocessor involvement, the talker/listener chip implements all interface functions, except controller performance, such as handling data transfers, handshake protocols, listener/talker address procedures, device clearing and triggering, service requests, and parallel and serial polling schemes (Fig 4).

Within the chip architecture are eight read (output) and eight write (input) registers. One input register holds the data that are to be moved from the bus to the microprocessor when a device is listening. An output register holds the data byte that is to be transferred to the bus when a device is ready to talk. The other seven write and seven read registers control various chip functions.

Interrupt status registers 1 and 2 store 12 different interrupt flags. For example, one bit in the Interrupt Status 2 register reflects changes in a device's addressed state. The microprocessor can poll both registers to determine which flag caused the interrupt, and can then branch to the appropriate service routine. Two corresponding interrupt mask registers allow designers to mask any interrupt. A serial poll status register holds device status information, and a serial poll mode register is available so that the microprocessor can verify this status. An address mode register contains a device's addressing mode, as determined by the microprocessor. An address status register monitors the address status (ie, active talker or active listener) of a device.

Two address registers store the assigned device addresses. An End-Of-Sequence (EOS) register contains a designer specified end of string code for delimiting data block transfers by flagging the last byte with EOI. A command pass-through register feeds non-GPIB commands to the microprocessor. An auxiliary mode register holds local messages to control reset, power on, etc.

Among the chip's capabilities are a programmable data transfer rate from 62k to 525k bytes/s, three addressing modes, and an EOS message recognition. With a programmable data transfer rate, the designer controls the handshake rate of the interface to match the data transfer rate to the devices on the bus.

The three addressing modes permit flexibility in designating talkers/listeners. The dual primary address mode, for example, allows both a talker and a listener address to be assigned to a device. With the primary/secondary address mode, multiple devices of the same type can have the same primary address, but a different secondary address. In the third addressing mode, devices can have both dual primary and dual secondary addresses.

Data block transfers are made easier with the EOS register. This register holds the character that signals an end-of-block transfer. When a data byte loaded into the data-out register matches the byte in the EOS register, the talker/listener chip asserts the EOI line, signaling an end of transfer.

Controller Chip Capabilities

The 8292 controller chip (Fig 5) implements the controller function of the Standard. In conjunction with the 8291, the controller forms a complete standard interface, including the capability of handling the transfer control protocol. This ability gives the designer an option to accommodate multiple controllers on a single bus.

Additionally, the 8292 performs all the tasks necessary in a complete controller design. It responds to

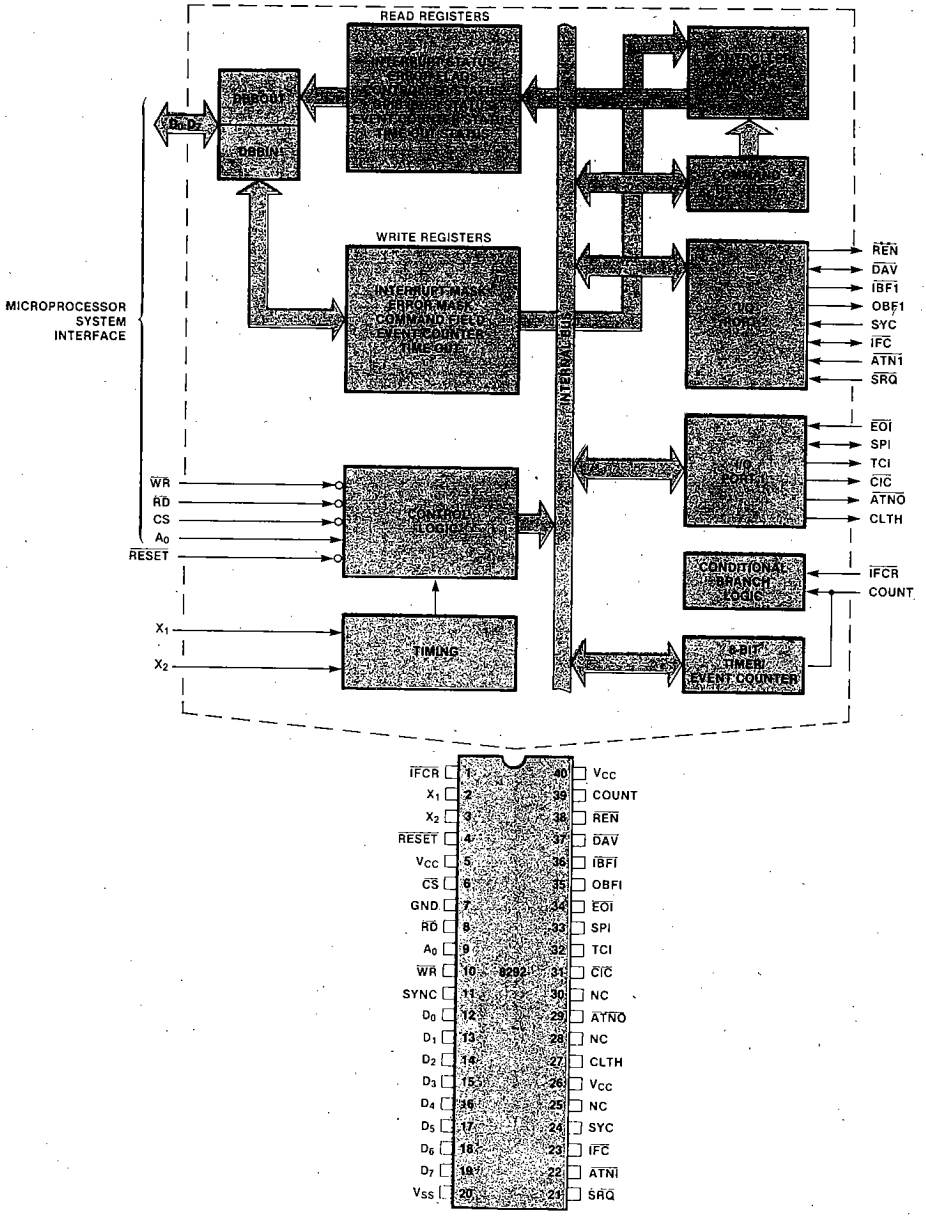


Fig 5 GPIB controller chip. 8292 chip works in conjunction with 8291 to perform GPIB controller interface functions. It implements local control commands from microprocessor according to IEEE Std 488 protocol. Additionally, it processes such inputs from bus as SRQ and EOI. Furthermore, it can send the full repertoire of GPIB control messages, including REN, IFC, ATN, and EOI

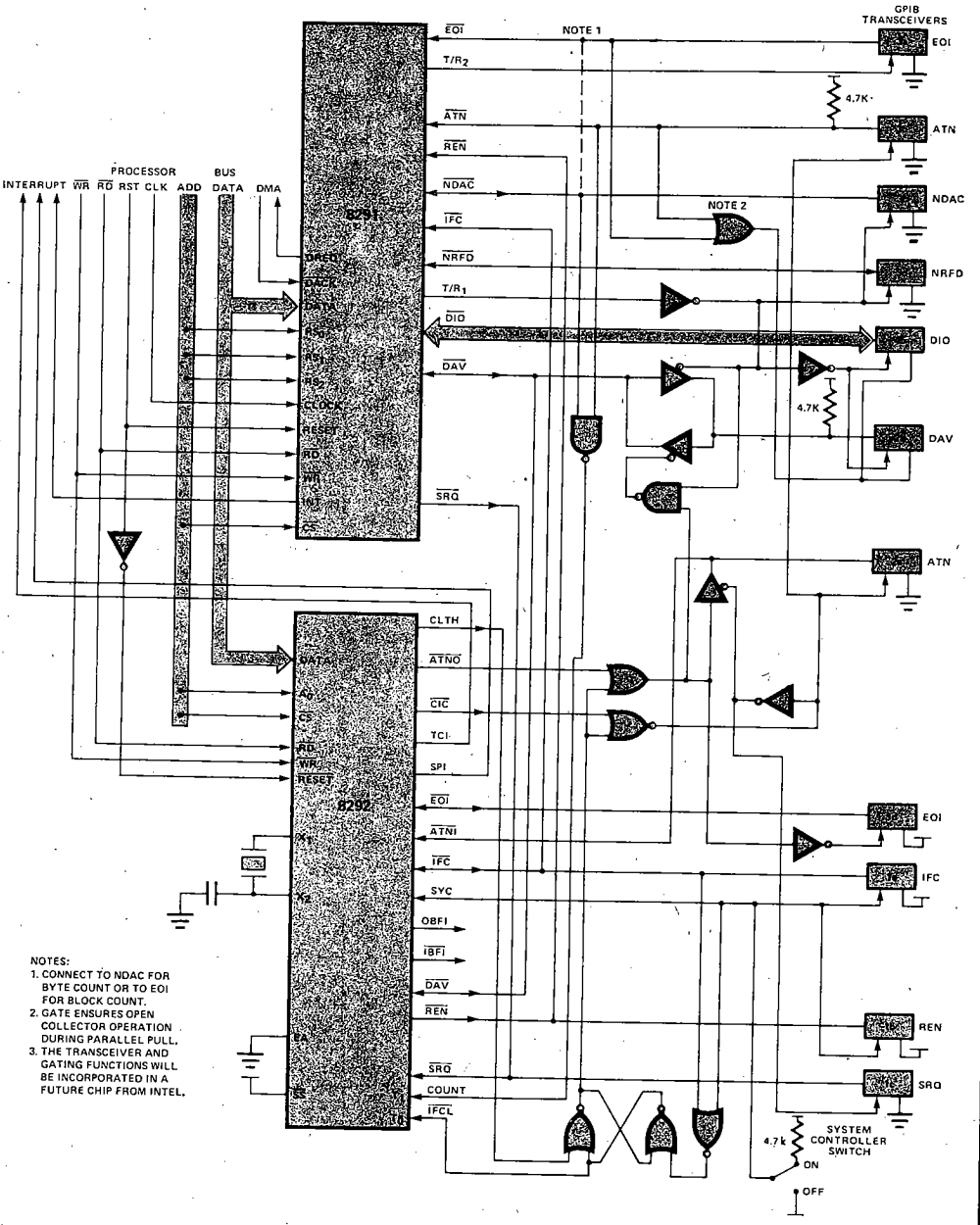


Fig 6 System configuration using chip set. In conjunction with 8291, 8292 performs complete controller function. Together with shared bus transceivers, chip set forms a complete IEEE Std 488 interface. In addition, DMA interface may be implemented through 8291 with 8237 DMA controller

service requests (SRQS), configures other devices on the bus for remote control by sending Remote Enable (REN), and sends Interface Clear (IFC), allowing for control seizure to reinitialize the bus. More importantly, the controller chip can take control of the bus synchronously with the handshake, preventing the destruction of any data transmission in progress.

Internally, the controller chip has 10 dedicated registers for programming and for monitoring status. Through the use of the Interrupt Status and Interrupt Mask registers, the designer can configure the controller to interrupt the microprocessor on selected events. An Event Counter and a corresponding status register are available to monitor and control either byte counts or block counts. A Time-Out register may be set by the designer to program a time-out error function; a corresponding status register contains the current value in the time-out counter. In conjunction with these registers, error control can be programmed with the Error Flags and Error Mask registers. Finally, Controller and GPIB Status registers are available. Each of these registers is read or programmed through a dedicated command buffer.

Chip Set Application

The talker/listener and controller chips connect to the standard interface bus through noninverting bus transceivers (Fig 6). These transceivers provide the 48-mA bus drive capability needed to meet the electrical portion of the IEEE Std 488 specification—not directly possible with existing metal oxide semiconductor (MOS) parts. The talker/listener chip can interface directly to microprocessor memory through a direct memory access (DMA) controller, such as an 8237.

The microprocessor drives the talker/listener with a short stored program (see Table), containing initialization conditions, such as data transfer rate, address mode, and other designer requirements. Microprocessor data handling is limited to taking bytes off the bus after they arrive or putting bytes of data on the bus. Interrupt service routines are necessary for each unmasked interrupt. Although 12 interrupts are available, not all have to be used. All other standard bus functions are handled by the 8291.

To send a byte of data, the microprocessor writes the byte into the talker/listener data-out register. The chip then transmits the data byte over the bus lines in conjunction with the handshake lines. Next, the NRFD line is checked to see if it is ready for data. If a ready for data message is detected, the talker/listener sends a DAV signal until it receives a data accepted message from the interface's NDAC line. The 8291 also generates a Byte Out (BO) interrupt, setting the BO flag in the interrupt status register. When its interrupt pin is activated, the microprocessor reads the interrupt status register and responds to the interrupt with an appropriate service routine.

The 8292 handles all hardware aspects of the controller function: SRQ input, ATN, IFC, EOI, and REN outputs. Meanwhile, the designer defined aspects of a given GPIB system are handled by processor software. For example, the processor is responsible for knowing which device on the bus corresponds to which device address. The processor then uses the 8291 to transmit coded Controller commands as the 8292 asserts ATN.

Table. Talker (Transmitter) Initialization

8291 Register	Data	Comments
5W	02H	Reset
4W	01H	Address Mode 1 (Primary-Secondary)—Note that talk only could be selected here, eliminating need for controller address.
6W	02H	Address 0 = 02H
6W	92H	Address 1 = 12H
0W	FFH	Enable All Interrupts
1W	3FH	Enable All Interrupts, DMA
5W	A4H	Enable High Speed Data Transfer
5W	00H	PON, Release Initialization State

Summary

Bus interface designs that previously required 150 or 200 MSI/SSI chips may now be implemented with a GPIB peripheral chip set. For designers, this hardware set means less design time and cost, resulting in increased reliability and versatility in IEEE Std 488 bus interfaces custom programmed for dedicated applications.

Bibliography

S. C. Baunach, "Design Advantages and Limitations in Connecting Computational and Readout Equipment to the GPIB," Western Electronic Show and Convention, Sept 1976

A. Kaminker and A. Menachem, "LSI Facilitates GPIB Implementation," *IEEE Proceedings on Microcomputer Based Instrumentation*, June 1978

D. C. Loughry and M. S. Allen, "IEEE Standard 488 and Microprocessor Synergism," *Proceedings of the IEEE*, Feb 1978

Ronald M. Williams is a product manager for peripheral controllers in Intel's Microcomputer Components Division. In addition to GPIB devices, he has been involved in introductions of dynamic RAM and CRT controllers. He holds a BS degree from Trinity College, an MS degree from Rensselaer Polytechnic Institute, and an MBA degree from the University of Chicago.